(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 738 059 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.10.1996 Bulletin 1996/42

(51) Int Cl.[6]: **H04L 12/26**

(21) Application number: **96302330.4**

(22) Date of filing: **02.04.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.04.1995 US 419213**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Finney, Damon W.**
  San Jose, California 95133 (US)
- **Rayfield, Michael James**
  Tucson, Arizona 85715 (US)

(74) Representative: **Lloyd, Richard Graham**
IBM (UK) Ltd,
UK Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and apparatus for testing links between network switches

(57) A test link protocol which continuously monitors each link in a network to ensure that the link is correctly transmitting data. Each switch, or torus has at least one of two functional components: Send Test and Receive Test. The Send Test component monitors control codes at a torus link output. The Receive Test component monitors control codes at a torus link input.

After a predetermined interval, the Send Test component makes a request to send a test_link control code. The torus sends the test_link code to the neighbouring torus, where it is removed from the data stream and sent to that torus' Receive Test. The Receive Test then generates a response message and makes a request to send that message back to the originating torus. After receiving the message, the Send Test analyzes the message to determine whether the network link is working correctly. An error is also declared if the Send Test does not receive a reply within a predetermined interval.

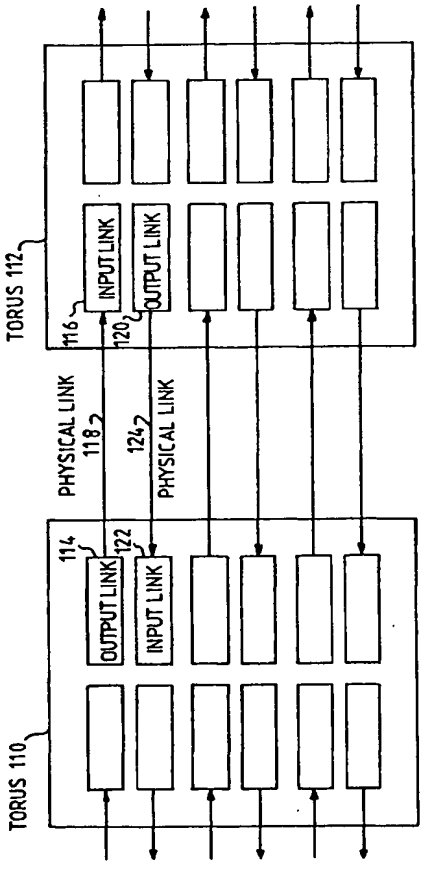

FIG. 1

EP 0 738 059 A2

Description

This invention pertains in general to data transmitting networks and more particularly to an error detection protocol for detecting errors between network switches.

Large scale data processing networks typically comprise many different switches separated by communication links. Each switch is linked to one or more other switches. These switches constantly receive and transmit data. Data transmission protocols are used to ensure that data passes through the network without error.

Data transmission protocols typically rely on control codes to ensure that all switches and links are functioning correctly. For example, a transmitting switch will send a code indicating that data has been transmitted and a receiving switch will send a reply acknowledging that data was received. The transmitting switch will expect to receive the acknowledgement within a pre-determined time period. Otherwise, a timeout error occurs.

Often, a timeout error is the first indication of network failure. Such an error, however, does not indicate what the error was or isolate the part of the network that failed. Thus, it is difficult for a switch or network administrator to determine which part of the network has failed. Moreover, switches waste valuable time waiting for a never-arriving acknowledgement code.

Therefore, it is an object of the present invention to provide a method and apparatus for determining when a network error has occurred, where a network error has occurred and to provide a method and apparatus for isolating the location of a network error.

This invention provides a network having an error detection protocol, said network comprising: a first switch; a second switch linked to said first switch; a send test component associated with said first switch for sending messages to said second switch and comparing messages received from said second switch with said send test component to detect network errors; and a receive test component associated with said second switch for responding to messages received from said send test component.

In at least a preferred embodiment, a test link protocol continuously monitors each link in the network to ensure that the link is correctly following a higher level protocol. Each torus, or switch, is connected to at least one other torus by a pair of uni-directional links, one sending link and one receiving link. Each sending link is connected to the receiving link of another torus and vice-versa. Each torus has at least one of two functional components: Send Test and Receive Test. The Send Test component monitors control codes at the torus link output. The Receive Test component monitors control codes at the torus link input. The test link protocol is implemented for each pair of links in the network and tests each pair separately.

After a predetermined interval, the Send Test component automatically makes a request to send a test_link control code. The torus sends the test_link code to the neighbouring torus, where it is removed from the data stream and sent to that torus' Receive Test. The Receive Test then generates a response message and makes a request to send that message back to the originating torus. After receiving the message, the Send Test analyzes the message to determine whether the network link is working correctly. If an error is detected, the Send Test issues an error message and shuts down the network link that the message used. An error message is also sent if the Send Test does not receive a reply within a predetermined interval.

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions of an embodiment thereof taken in conjunction with the accompanying drawings, in which:

Figure 1 depicts a typical configuration of a torus-switch link interconnection;

Figure 2 depicts a detailed view of a pair of input/output links connecting two tori;

Figure 3 depicts a sample state machine for tracking control codes;

Figure 4 depicts a detailed view of the logical blocks in the Send Test component;

Figure 5 depicts a state machine used to determine the maximum count of the Send Test counter;

Figure 6 depicts a state machine used to control the behaviour of the Send Test component; and

Figure 7 depicts a detailed view of the logical blocks in the Receive Test component.

The present invention can be implemented in any network in which multiple switches are connected by logical data channels. For example, the invention can be implemented in a large network having thousands of switches and connections or in a small network connecting computer peripherals to the system bus of a computer system.

Figure 1 shows a typical configuration of interconnected tori. A "torus" is a cross-point switch used to transmit information across a network. Two tori are shown 110, 112. Torus 110 has an Output Link 114 connected to torus 112's Input Link 116 via physical link 118. Likewise, torus 112's Output Link 120 is connected to torus 110's Input Link 122 via physical link 124.

A pair of interconnected Input/Output Links 114, 116, 120, 122 forms a logical data channel. The links themselves 118, 124 are herein referred to as "physical links," but may in fact be based upon any form of data transmission. Thus, the Input/Output Links 114, 116, 120, 122 need not be physically connected. Each torus

110, 112 sends messages to the other torus via its Output Link 114, 120. The messages follow a specific protocol. The protocol, in turn, is composed of control codes. The control codes provide information such as the size of the message or what action the receiving torus is to perform on the message.

Figure 2 shows a detailed view of a pair of Input/Output links connecting the tori 110, 112 of Figure 1. Also shown are the Send 210 and Receive 212 Test functional components. These two components 210, 212 are present for every pair of links 118, 124 between tori. The Send 210 and Receive 212 Test functional components implement the test link protocol between the tori.

Each Output Link 114, 120 sends data from its respective torus 110, 112 over a physical link 118, 124 to the Input Link 116, 122 of the other torus 112, 110. Output Links 114, 120 also process and act on requests from the Send 210 and Receive 212 Test components to send messages to the other torus.

Input Links 116, 122 receive data from their respective physical links 118, 124. In addition, the Input Links 116, 122 detect certain control codes, remove those codes from the data streams, and then notify either Send 210 or Receive 212 Test of the codes' arrival. For example, Input Link 116 detects and removes test_link control codes from the data stream and notifies Receive Test 212 of their arrival. Further operation of the Input 116, 122 and Output Links 114, 120 is described below.

Send Test 210 is connected to torus 110 such that it can monitor messages sent over physical link 118, send messages to Output Link 114 and receive messages from Input Link 122. Send Test 210 has three main functions: monitor control codes; determine when to send a test_link message to Receive Test 212; and compare the state of Receive Test 212 to its own state.

Send Test 210 monitors control codes sent by Output Link 114 over physical link 118. Send Test 210 uses a state machine to track these codes. The actual configuration of the state machine is dependant upon the protocol used by the network. Since the state machine will be used for error detecting, however, both Send Test 210 and Receive Test 212 must use identical state machines.

Figure 3 shows a sample state machine for tracking control codes. This state machine has three possible states: "idle" 310; "control code one" 312; and "control code two" 314. The state machine transitions to each state when it detects the corresponding control code. For example, the state changes from "idle" 310 to "control code one" 312 when Output Link 114 sends a control_code_1 over physical link 118.

Send Test 210 determines when to send a test_link message to Receive Test 212. The messages can be sent after any predetermined time interval, but the best mode is described herein. If no message traffic is on physical link 118, Send Test 210 requests that Output Link 114 send a test_link every 65,536 clock cycles. If a message is on physical link 118, Send Text 210 requests a test_link 64 cycles after the beginning of the message. Output Link 114 sends the test_link message to Receive Test 212 as soon as traffic permits.

Receive Test 212 has two main purposes: monitor control codes carried by physical link 118 and send the current status of Receive Test 212's state machine to Send Test 210 in response to a test_link message.

Receive Test 212 is connected to physical link 118 such that it can monitor control codes sent on the link. Receive Test implements a state machine like that shown in Figure 3 and updates it according to the control codes it detects.

When Input Link 116 receives a test_link message, it notifies Receive Test 212 of the message. In response, Receive Test 212 formulates a rcv_test_response message containing the current state of the Receive Test 212 state machine. Next, Receive Test 212 requests that Output Link 120 send the rcv_test_response message. Output Link 120 sends the rcv_test_response message to Input Link 122 via physical link 124.

Send Test 210 then compares the state of Receive Test 212 with that of itself. In normal operation, both test components 210, 212 should be in the same state because they implement the same state machine. If the components 210, 212 are in different states, a network error has occurred. In addition, an error occurs if a rcv_test_response message is not received by Send Test 210 within a predetermined time interval.

Send Test

Figure 4 depicts a detailed view of the logical blocks in Send Test 210. A decoder 410 and a counter 412 are connected to physical link 118. Decoder 410 is connected to a Control Code Monitoring State Machine ("CCMSM") 414, which, in turn, is connected to a comparator 416. Counter 412 is connected to a Send Test Link State Machine ("STLSM") 418. A system clock, which generates clock cycles, is also present but not shown.

Decoder 410 monitors and decodes control codes sent over physical link 118. The control codes are then sent to CCMSM 414. CCMSM 414 implements a state machine like that shown in Figure 3.

Counter 412 counts clock cycles. The limit of Counter 412 is determined using the state machine shown in Figure 5. If a message has been sent on physical link 118, Counter 412's maximum count is 64 cycles from the beginning of the message. If STLSM 418 has requested that a test_link control code be sent, Counter 412's maximum count is 32. Otherwise, Counter 412's maximum is 65,536. When Counter 412 reaches its maximum count, it sends a signal to STLSM 418.

STLSM 418 determines when to request that a test_link message be sent and whether a network error has occurred. STLSM 418 implements the state machine shown in Figure 6. The state machine has four states: "increase counter" 610; "request test_link" 612; "test response" 614; and "compare states" 614.

STLSM 418's state machine remains at "increase counter" state 610 until it receives a counter_at_limit signal from counter 412. Then, the state machine transitions to "request test_link" state 612. At state 612, STLSM 418 sends a signal to Output Link 114 requesting that a test_link message be sent over physical link 118. Also at state 612, STLSM 418 sets Counter 412 to 32. Then, STLSM 418 transitions to "test response" state 614.

At state 614, the state machine waits for either a rcv_test_response signal from Input Link 122 or a counter_at_limit signal from counter 412. If a rcv_test_response signal is received, the state machine transitions to "compare states" state 614 and then back to "increase counter" state 610. A counter_at_limit signal received while at state 614 indicates that 32 cycles have elapsed since the test_link message was sent. Therefore, an error has occurred because a response from torus 112 was not received within 32 cycles.

Comparator 416 compares the state of Receive Test 212 with that of Send Test 210. Comparator 416 receives signals from Input Link 122 and from CCMSM 414. The state of Receive Test 212 is embedded in the rcv_test_response signal received from Input Link 122. Using these two signals, comparator 416 compares the state of Receive Test 212 with the state of CCMSM 414. If the states are different and STLSM 418 is in the compare states state 614, then a control code has been lost and a network error has occurred.

Receive Test

Figure 7 depicts a detailed view of the logical blocks in Receive Test 212. A decoder 710 is connected to physical link 118. Decoder 710 is also connected to a CCMSM 712. Input Link 116 is connected to a Control Code Test Link Decoder ("CCTLD") 714. CCTLD 714, in turn, is connected to a latch 716. Both CCMSM 712 and latch 716 have outputs 718, 720 connected to Output Link 120 (not shown in Figure 7). A system clock, which generates clock cycles, is also present but not shown.

Decoder 710 monitors and decodes control codes sent over physical link 118. The decoded codes are then sent to CCMSM 712. CCMSM 712 implements a state machine like that of Figure 3. In addition, CCMSM 712 has an output 718 indicating its current state. This output 718 is connected to Output Link 120.

CCTLD 714 decodes test_link signals received by Input Link 116. A test_link signal lasts for only one cycle and, thus, so does CCTLD 714 's output signal. Therefore, CCTLD 714 sends its output to latch 716. Latch 716's output creates a one cycle pulse which requests that Output Link 120 send a rcv_test_response message over physical link 124. When traffic permits, Output Link 120 sends a rcv_test_response message containing the state of control code monitoring state machine 712.

The rcv_test_response message is received by Input Link 122. Input Link 122 detects and removes the message from the data stream and passes its contents to Send Test 210. As discussed above, Send Test 210 then uses the message to compare the states of Send Test 210 and Receive Test 212.

When Send Test 210 determines that a network error has occurred, it will normally shut down the logical communication channel on which the error occurred or perform another predetermined action such as reset the logical channel or notify a network supervisor. Of course, other channels can still be used to send messages through the network.

There has been described a test link protocol which continuously monitors each link in a network to ensure that the link is correctly transmitting data. Each switch, or torus has at least one of two functional components: Send Test and Receive Test. The Send Test component monitors control codes at a torus link output. The Receive Test component monitors control codes at a torus link input.

After a predetermined interval, the Send Test component makes a request to send a test_link control code. The torus sends the test_link code to the neighbouring torus, where it is removed from the data stream and sent to that torus' Receive Test. The Receive Test then generates a response message and makes a request to send that message back to the originating torus. After receiving the message, the Send Test analyzes the message to determine whether the network link is working correctly. An error is also declared if the Send Test does not receive a reply within a predetermined interval.

Although the present invention and its advantages have been described in detail, it should be understood that various changes, substitutions and alterations can be made herein without departing from the scope of the invention as defined by the appended claims.

Claims

1. A network having an error detection protocol, said network comprising:

   a first switch;

   a second switch linked to said first switch;

   a send test component associated with said first switch for sending messages to said second switch and comparing messages received from said second switch with said send test component to detect network errors; and

   a receive test component associated with said second switch for responding to messages received from said send test component.

2. A network as claimed in claim 1, further comprising:
a counter, said counter associated with said send test component, for determining when to send said messages to said second switch and to determine when a reply should be received from said second switch.

3. A network as claimed in claim 1 or claim 2, further comprising:

a first state associated with said send test component; and

a second state associated with said receive test component, wherein said receive test component sends said second state to said first switch and said send test component compares said second state with said first state to detect said network errors.

4. A network as claimed in any preceding claim, wherein said first and second switches are torus switches.

5. A network as claimed in any preceding claim, further comprising:
error handling means for controlling said first and second switches when said network errors are detected.

6. A network as claimed in claim 5, wherein said error handling means further comprises:
means for shutting down said network when said network errors are detected.

7. A network as claimed in claim 5, wherein said error handling means further comprises:
means for resetting said network when said network errors are detected.

8. A network as claimed in claim 5, wherein said error handling means further comprises:
means for notifying a network supervisor when said network errors are detected.

9. A network as claimed in any preceding claim, further comprising:

a first input/output link associated with said first switch, said first input/output link having a first input link and a first output link; and

a second input/output link associated with said second switch, said second input/output link having a second input link and a second output link, said first output link connected to said second input link by a uni-directional connection and said second output link connected to said first input link by a uni-directional connection.

10. A network as claimed in claim 9, wherein said uni-directional connections are physical links.

11. A method of error detection in a network having a first switch and a second switch, comprising the steps of:

sending a message from said first switch to said second switch;

determining, in response to said message, a state of a receive test component associated with said second switch; and

replying to said message by sending said state of said receive test component to said first switch; and

comparing said state of said receive test component with a state of a send test component associated with said first switch to detect a network error.

12. A method as claimed in claim 11, wherein said message is sent to said second switch after a predetermined time interval.

13. A method as claimed in claim 11 or claim 12, further comprising the step of:
detecting said network error if said reply is not received by said first switch within a predetermined time interval.

14. A network switch arranged to operate in a network using a method as claimed in any of claims 11 to 13.

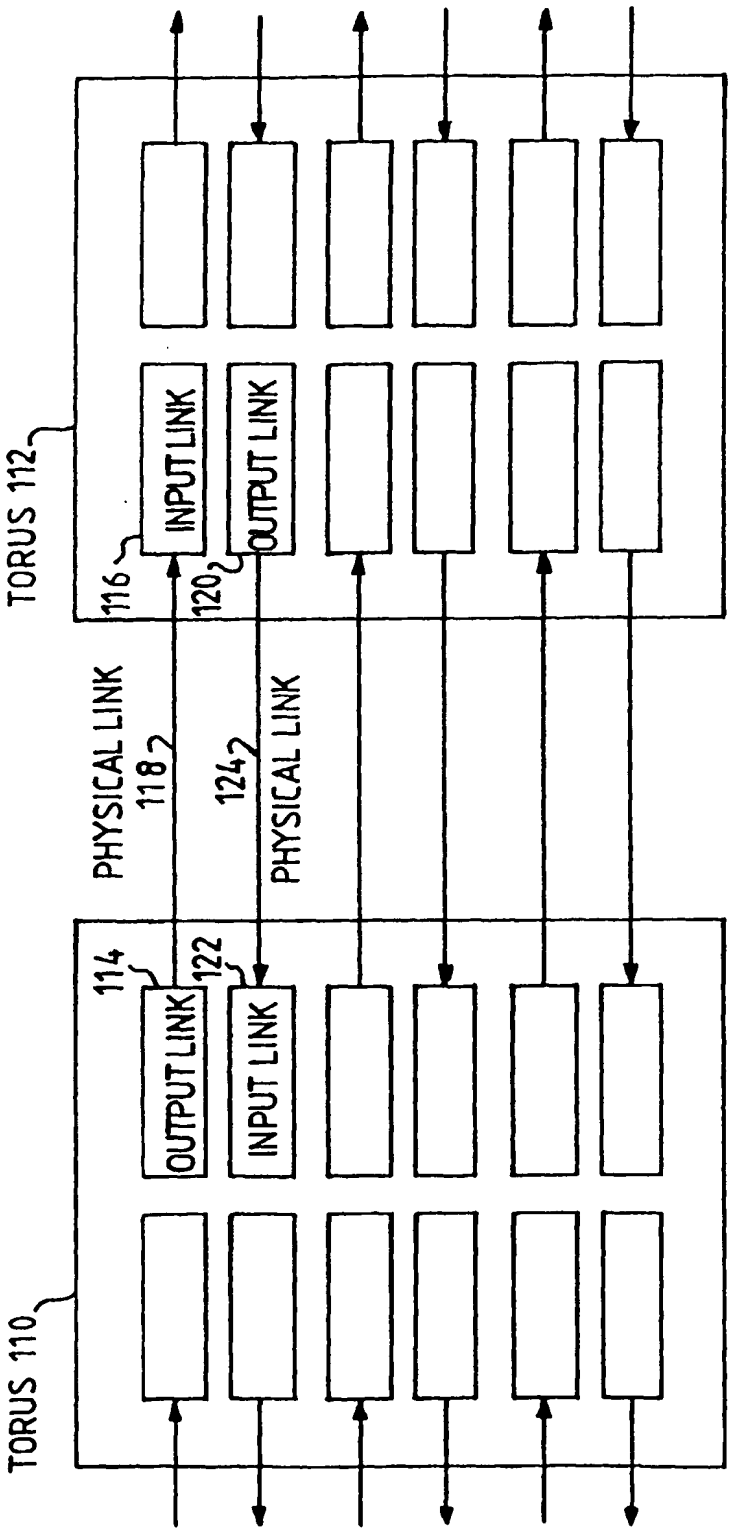
TORUS 112
INPUT LINK
OUTPUT LINK
116
120
PHYSICAL LINK
118
PHYSICAL LINK
124
114
122
OUTPUT LINK
INPUT LINK
TORUS 110

FIG. 1

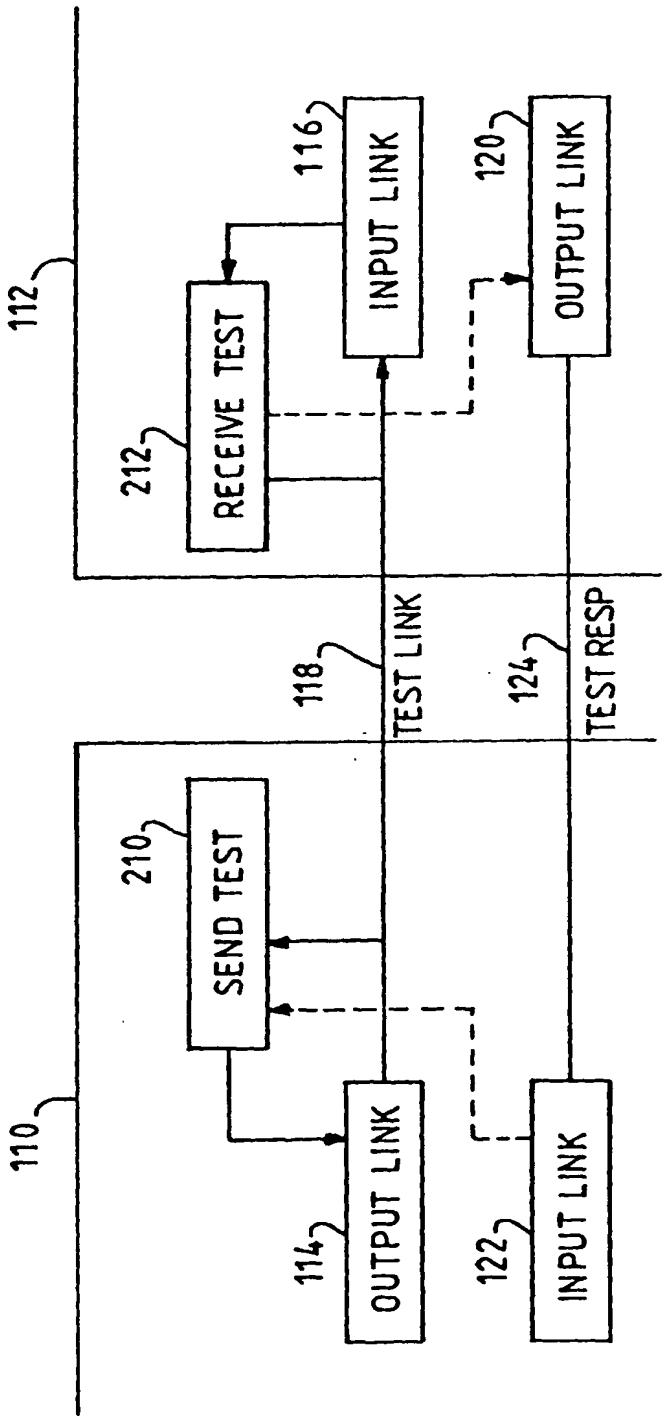
112
212
RECEIVE TEST
116
INPUT LINK
120
OUTPUT LINK
118
TEST LINK
124
TEST RESP
110
210
SEND TEST
114
OUTPUT LINK
122
INPUT LINK

FIG. 2

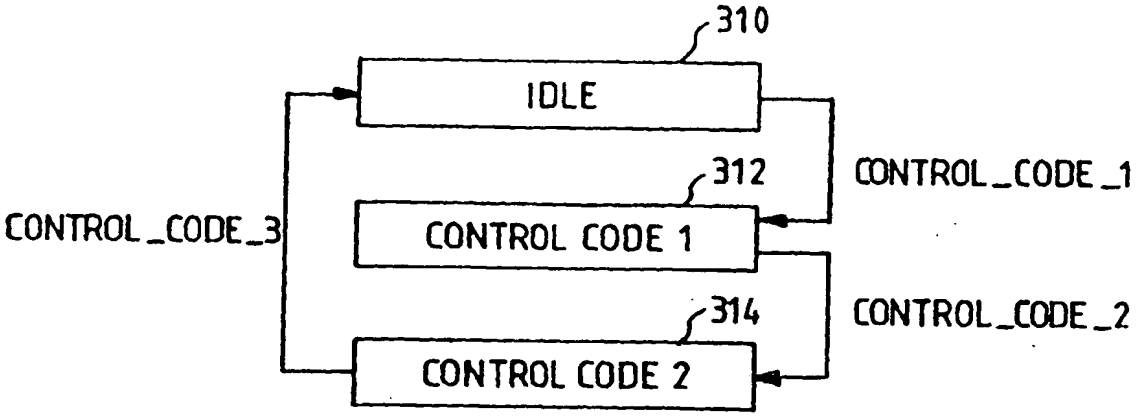
310
IDLE
312
CONTROL_CODE_1
CONTROL_CODE_3
CONTROL CODE 1
314
CONTROL_CODE_2
CONTROL CODE 2

FIG. 3

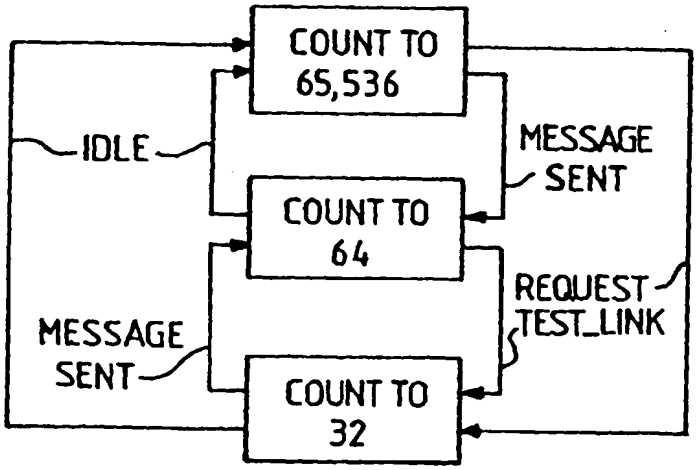
COUNT TO
65,536
IDLE
MESSAGE
SENT
COUNT TO
64
REQUEST
TEST_LINK
MESSAGE
SENT
COUNT TO
32

FIG. 5

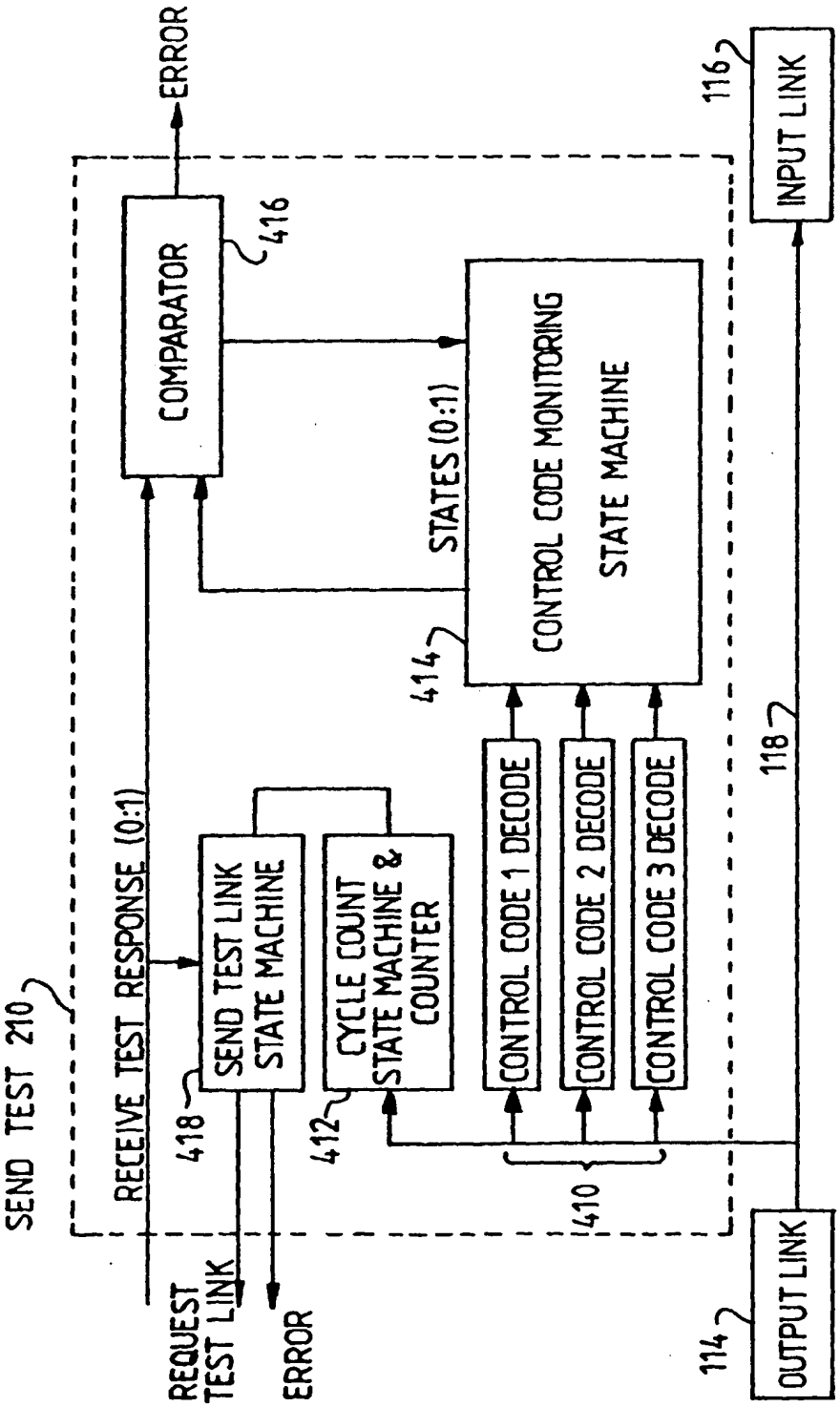
SEND TEST 210
RECEIVE TEST RESPONSE (0:1)
REQUEST TEST LINK
ERROR
SEND TEST LINK STATE MACHINE
418
CYCLE COUNT STATE MACHINE & COUNTER
412
CONTROL CODE 1 DECODE
CONTROL CODE 2 DECODE
CONTROL CODE 3 DECODE
410
COMPARATOR
416
ERROR
STATES (0:1)
CONTROL CODE MONITORING STATE MACHINE
414
INPUT LINK
116
OUTPUT LINK
114
118

FIG. 4

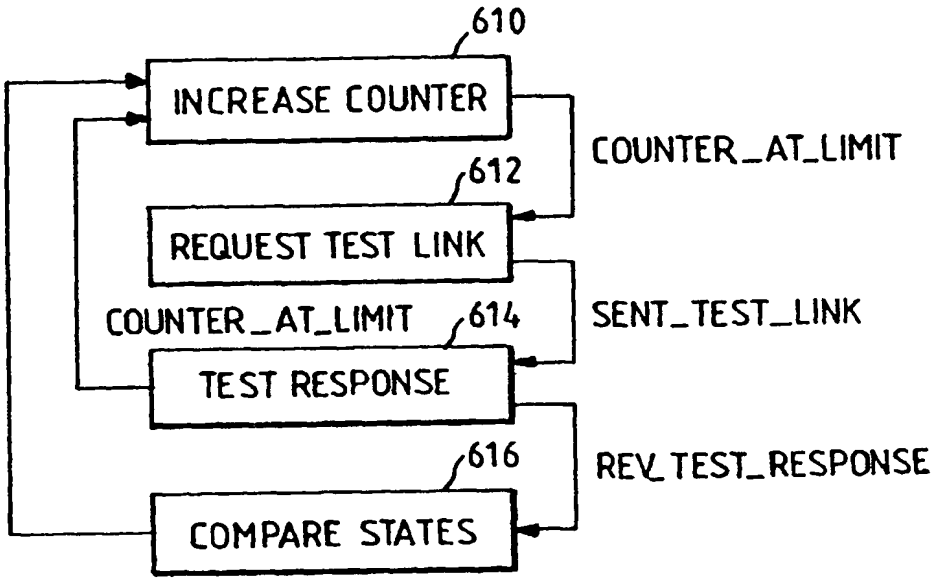
610
INCREASE COUNTER
COUNTER_AT_LIMIT
612
REQUEST TEST LINK
SENT_TEST_LINK
COUNTER_AT_LIMIT
614
TEST RESPONSE
REV_TEST_RESPONSE
616
COMPARE STATES

FIG. 6

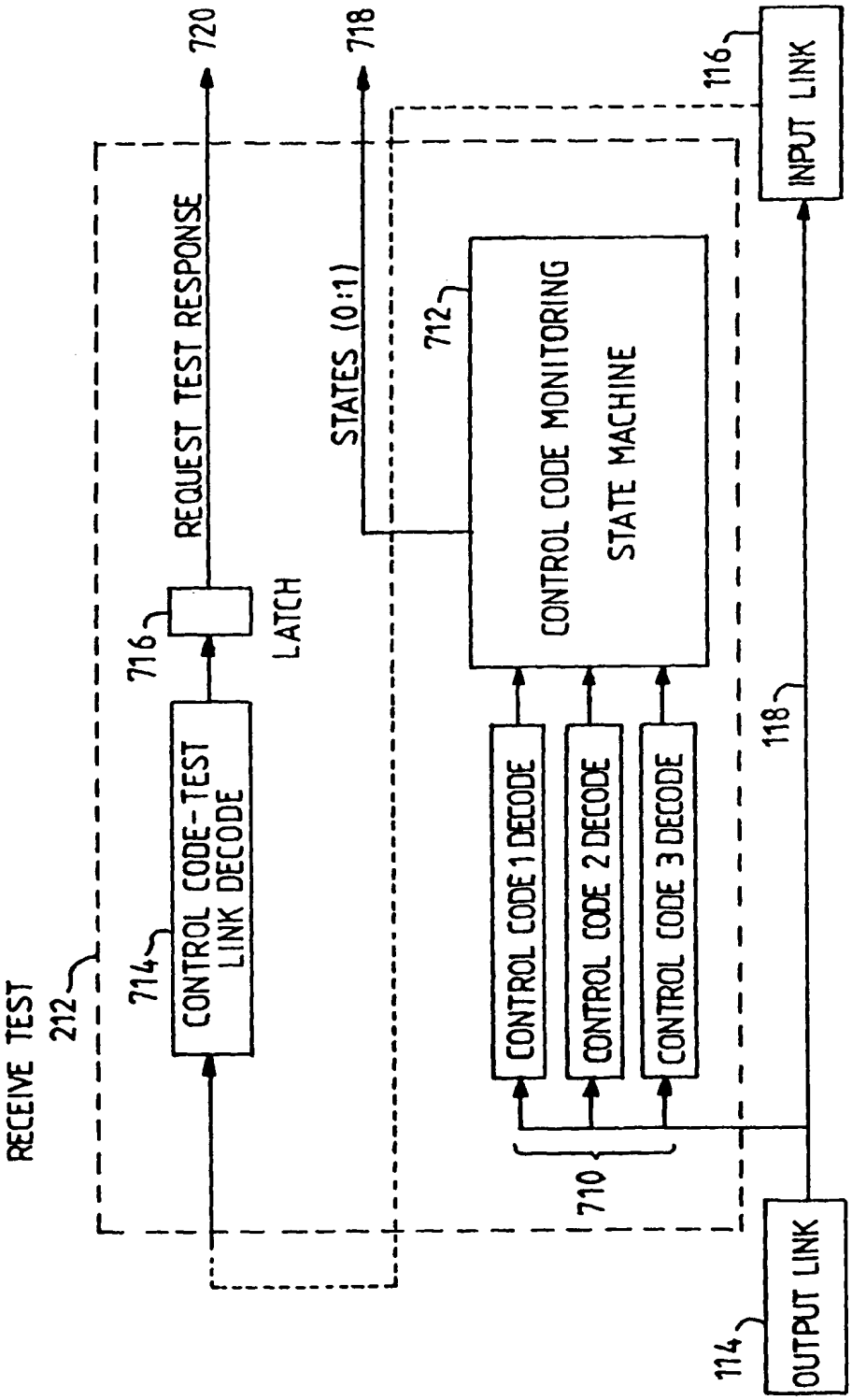
RECEIVE TEST
212
714
CONTROL CODE-TEST LINK DECODE
716
LATCH
REQUEST TEST RESPONSE
720
STATES (0:1)
718
712
CONTROL CODE MONITORING STATE MACHINE
CONTROL CODE 1 DECODE
CONTROL CODE 2 DECODE
CONTROL CODE 3 DECODE
710
116
INPUT LINK
118
114
OUTPUT LINK

FIG. 7

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 738 059 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
19.02.1997 Bulletin 1997/08

(43) Date of publication A2:
16.10.1996 Bulletin 1996/42

(21) Application number: **96302330.4**

(22) Date of filing: **02.04.1996**

(51) Int Cl.[6]: **H04L 12/26**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.04.1995 US 419213**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Finney, Damon W.**
  San Jose, California 95133 (US)
- **Rayfield, Michael James**
  Tucson, Arizona 85743 (US)

(74) Representative: **Lloyd, Richard Graham**
IBM (UK) Ltd,
UK Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and apparatus for testing links between network switches

(57) A test link protocol which continuously monitors each link in a network to ensure that the link is correctly transmitting data. Each switch, or torus has at least one of two functional components: Send Test and Receive Test. The Send Test component monitors control codes at a torus link output. The Receive Test component monitors control codes at a torus link input.

After a predetermined interval, the Send Test component makes a request to send a test_link control code. The torus sends the test_link code to the neighbouring torus, where it is removed from the data stream and sent to that torus' Receive Test. The Receive Test then generates a response message and makes a request to send that message back to the originating torus. After receiving the message, the Send Test analyzes the message to determine whether the network link is working correctly. An error is also declared if the Send Test does not receive a reply within a predetermined interval.

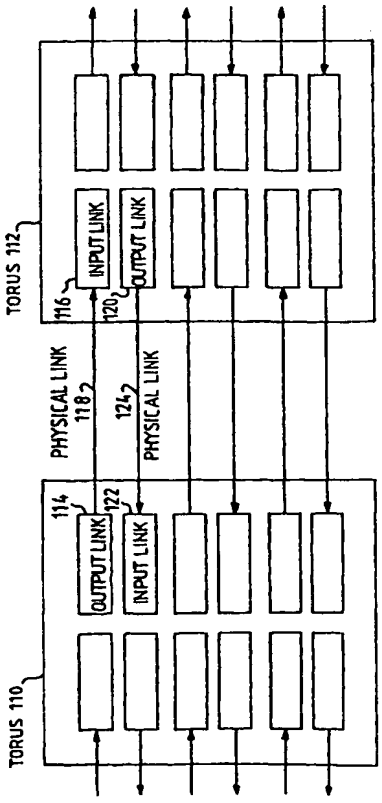

FIG. 1

EP 0 738 059 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 30 2330

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X Y	EP 0 085 437 A (NIPPON ELECTRIC KK) 10 August 1983 * abstract * * figures 2,3,5 * * page 2, line 4 - line 22 * * page 4, line 15 - line 26 * * page 8, line 9 - line 15 * ---	1-5,8, 11-14 9,10	H04L12/26
X	WO 92 22967 A (HEWLETT PACKARD) 23 December 1992 * abstract * * page 5, line 16 - line 22 * * page 9, line 11 - line 17 * * claim 14 * ---	1,2	
X	EP 0 510 822 A (TEXAS INSTRUMENTS INC.) 28 October 1992 * abstract * * page 2, line 43 - line 51 * * page 3, line 3 - line 19 * ---	1,5-7	
Y	EP 0 324 886 A (IBM) 26 July 1989 * figure 7 * * column 7, line 47 - line 51 * -----	9,10	TECHNICAL FIELDS SEARCHED (Int.Cl.6) H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	20 December 1996	Cichra, M

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.